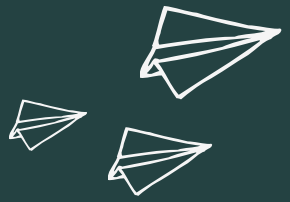


go 语 言 编 程

目录页

第六章
Part six

6.1 结构体

结构体(struct)是由一系列具有相同类型或不同类型的数据构成的集合。这些数据称为结构体的成员。

Go语言的结构体和其他语言的类有同等的地位，但Go语言放弃了包括继承在内的大量面向对象特性，只保留了组合这个最基础的特性。这也体现了Go语言的简洁性。

第六章 结构体、方法和接口

6.1.1 结构体定义

定义一个结构体相当于定义了一个新的类型。定义结构体的一般形式：

```
type 类型名称 struct {
    成员列表
}
```

例如，定义一个类型Person

```
type Person struct {
    name string
    age  int
}
```

需要注意的是，结构体成员的输入顺序也有重要的意义。如果成员的顺序不同，意味着定义了不同的结构体类型。一个命名为S的结构体类型将不能再包含S类型的成员，因为一个聚合的值不能包含它自身。 但是可以包含自身类型的指针。

第六章 结构体、方法和接口

6.1.2 结构体初始化

1> 按顺序直接初始化

2> 按照字段:值的方式初始化，这种方式的初始化对于字段的初始化的顺序可以是任意的。

3> 也可以先定义结构体变量，随后赋值。

```
p1 := Person{"张三", 20} // 1 按顺序初始化
fmt.Println(p1)
p2 := Person{age: 18, name: "李四"} // 2 采用字段:值的方式初始化，可以
                                        任意顺序
fmt.Println(p2)
p3 := Person{}// 3 未显示初始化，其成员默认被初始化为零值
fmt.Println(p3)
p3.name = "王五" //给每个成员赋值
p3.age = 19
fmt.Println(p3)
```

6.1.3 匿名组合实现继承

1 >匿名组合的实现

只声明一个成员对应的数据类型而不指明成员的名字,这类成员就叫匿名成员。通过匿名成员实现的匿名组合，可以完成继承的功能。

2>匿名组合的重名问题

匿名组合重名时，通过成员的数据类型区分不同类型中的成员。

6.1.4 匿名结构

结构体类型没有名称，定义结构体类型时直接初始化。

6.1.5 结构体比较

如果结构体的全部成员都是可以比较的，那么结构体也是可以比较的。可比较的结构体类型和其它可比较的类型一样，可以用于map的key类型。

6.2 方法

方法就是与某个类型(也可以是内置类型)关联的函数。理所当然我们可以给一个具体的结构体类型添加方法，使得它更像面向对象中的类。

6.2.1 方法声明

函数声明时，在它的名字之前加一个变量，即是一个方法。相当于为这种类型定义了一个独占的方法。方法可以被声明到任意类型，只要这个类型本身不是一个指针或接口。因为只有类型和指向它们的指针才可以是接收器。如果一个类型名本身是一个指针的话，是不允许出现在接收器中的。

第六章 结构体、方法和接口

方法声明的一般格式如下：

```
func  (变量名  类型)方法名称( [形参列表] ) [返回值列表]{
    方法体
}
或
func  (变量名  *类型)方法名称( [形参列表] ) [返回值列表]{
    方法体
}
```

在声明一个方法的接收者该是指针还是非指针类型时，需要考虑两方面：

1>对象本身是否特别大，如果声明为非指针变量时，调用会产生一次拷贝。

2>如果使用指针类型，需要注意这种指针类型指向的始终是一块内存地址，就算你对其进行了拷贝。

6.2.2 方法值和方法表达式

某个对象的方法称为方法值，某个类型的方法称为方法表达式。通过方法表达式调用方法时，需要将第一个参数作为接收器。

6.2.3 可见性

Go语言控制可见性的手段只有一种，首字母大小写。而且，可见性是包一级的而不是类型一级的。　名字的首字母大写，包外可见。首字母小写，只能在包内访问。

6.3 接口

6.3.1接口类型

接口类型是对其它类型行为的抽象和概括，因为接口类型不会和特定的实现细节绑定在一起。很多面向对象的语言都有的接口概念，但是Go语言中接口类型的独特之处在于它是满足隐式实现的。另外，一个接口也可以嵌入其它接口，即接口组合。

6.3.2 实现接口的条件

一个类型如果拥有一个接口需要的所有方法，那么这个类型就实现了这个接口。

6.3.3 接口赋值

1> 将一个对象赋值给一个接口

2> 将一个接口赋值给另一个接口

在Go语言中，只要两个接口拥有相同的方法列表（次序不同不要紧），那么这两个接口是等价的，可以相互赋值。

接口赋值并不要求两个接口必须等价。如果接口A的方法列表是接口B的方法列表的子集，那么接口B可以赋值给接口A。

6.3.4 接口查询

接口查询语法类似x.(T)，x表示一个接口，T表示另一个接口类型。用来判断接口x操作的对象的是否实现了另一个接口类型。

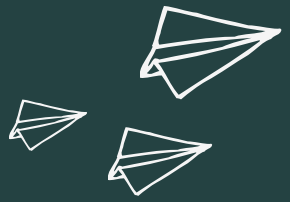
6.3.5 空接口

由于Go语言中任何对象实例都满足空接口interface{}，所以interface{}看起来像是可以指向任何对象的类型。一个函数把interface{}作为参数，那么他可以接受任意类型的值作为参数，如果一个函数返回interface{},那么也就可以返回任意类型的值。

6.3.6 类型查询

在Go语言中，还可以更加直截了当地询问接口指向的对象实例的类型。

```
//伪代码如下
var v1 interface{} = ...
switch v := v1.(type) {
    case int: // 现在 v 的类型是 int
    case string: // 现在 v 的类型是 string
    ...
}
```



THANKS

感谢聆听，期待反馈